```
/*
Method Hiding, Overriding and overloading
=========================================
 - Defining a 'static' method in super-class and in sub-class with the
   same signature, method name and parameters, is called method hiding

 - Defining a 'non-static' method in super-class and in sub-class with the
   same signature, method name and parameters, is called method overriding

 - Defining multiple 'static or non-static' method with the same name and different
   parameters 'type or list or order' is called method overloading

 - If the super-class given implementation logic is not satisfying subclass
   requirement then we must hide or override that super class method in subclass
   with the sub-class required logic. We must not add subclass required other logic
   directly in a super class method because it will affect other sub-classes.

 - For performing same operation with a different type or list or order of
   inputs we must overload method

 - In case of method hiding and method overriding
      we 'redefine' super-class method in sub-class
          to provide new implementation according to
          subclass requirement.

                  hence we can say in a method hiding and overriding,
                  the method in sub class and super class is a same method
                  with diff implementation

 - In case of method overloading
      we are 'defining' multiple methods with the same name with
          different parameter 'type or list or order' for performing
          same operation with different inputs.

                  hence we can say in method overloading,
                  the method in sub class and super class is a different method
                  with same name and different parameters.

 - Rule: Hence we can say
        1. We cannot hide or override a method in the same class,
           because both methods signature will be same, it is consider as
           duplicate method , compiler will throw error

        2. We can overload a method in either in the same class or in sub class,
           because both methods are differnt because their signature is different

Below programs shows hiding, overriding and overloading a method

*/

        class A {
          //hidden method
                static void m1() {
                        System.out.println("A SM m1");
                }

         //overridden method
                void m2() {
                        System.out.println("A NSM m2");
                }

         //Overloaded method
                static void m3() {
                        System.out.println("A SM m3");
                }

         //overloading method
                static void m3(int i) {
                        System.out.println("A SM m3(int)");
                }

        }
        class B {
          //method hidding
                static void m1() {
                        System.out.println("A SM m1 in class B");
                }

          //method overriding
                void m2() {
                        System.out.println("A NSM m2 in class B");
                }

          //overloading method
                static void m3(String s) {
                        System.out.println("B SM m3(String)");
                }
        }

        class C {
                static void m1() { }
                static void m1() { } //hiding in same class, leads CE

                void m2() { }
                void m2() { } //overriding in same class, leads CE

                void m3() { }
                void m3(int a) { } //overloaing in same class, no CE
        }

/*
Execution flow of hidden, overridden and overloaded methods
===========================================================

Static method, hidden method execution flow
===========================================
 - when we call super class SM by using super class type referenced
   variable by storing sub class object, it is executed only super class
   by not from sub class even though it is hiding in sub class
   because JVM does not have sub class information in SM

 - Compiler and JVM both search and execute SM, hidden method,
   in referenced variable type class
        A a1 = new B();
        a1.m1();  //A m1 (executed fom class A not from class B)

        - compiler and JVM serarch for m1() method
          in class A but not class B even though
          the object is class B and even if it is
          redefined in sub class B
```